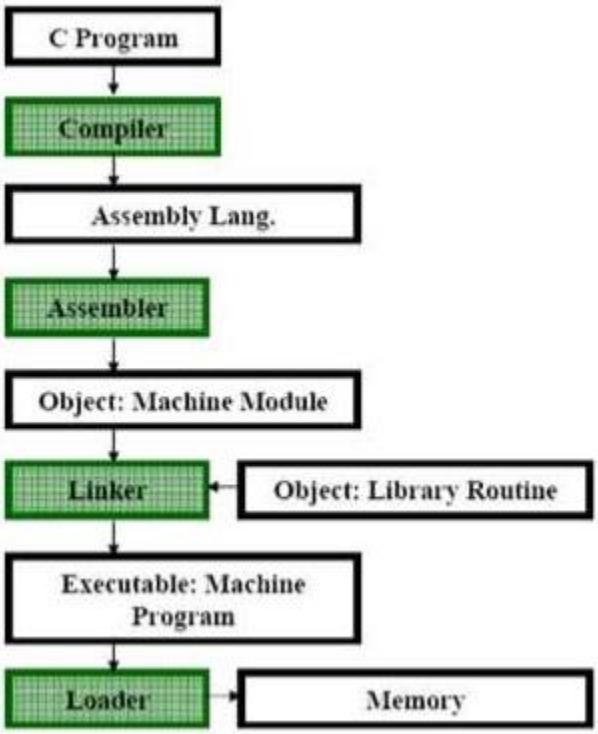
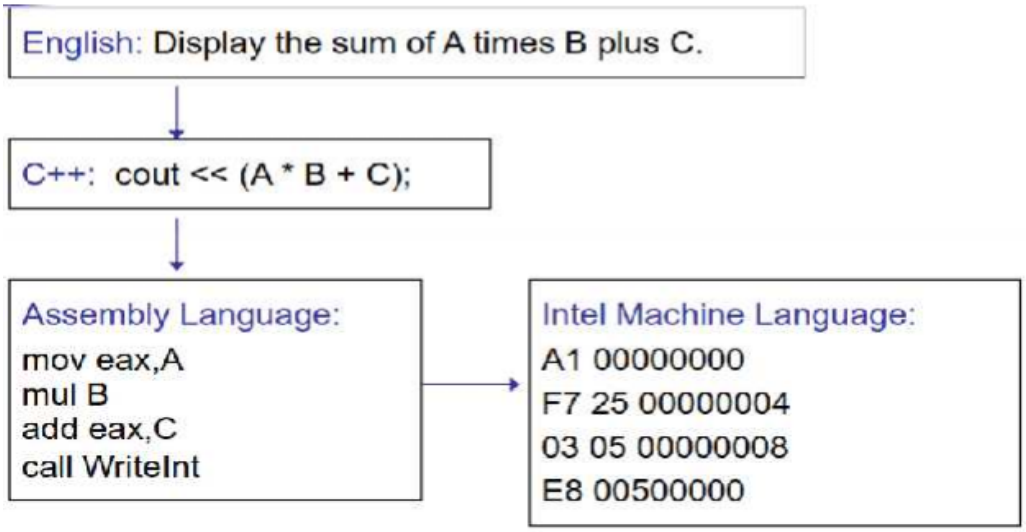
ASSEMBLY LANGUAGE LAB 01

Topic that is covered in this Lab:

1. **Introduction to Assembly Language**
2. **Benefits of Assembly**
3. **Why talking this course**
4. **Number System**
5. **Debug Introduction**



For Example:

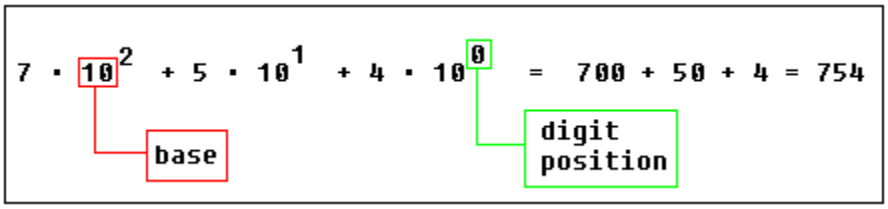


ASSEMBLY LANGUAGE LAB 01

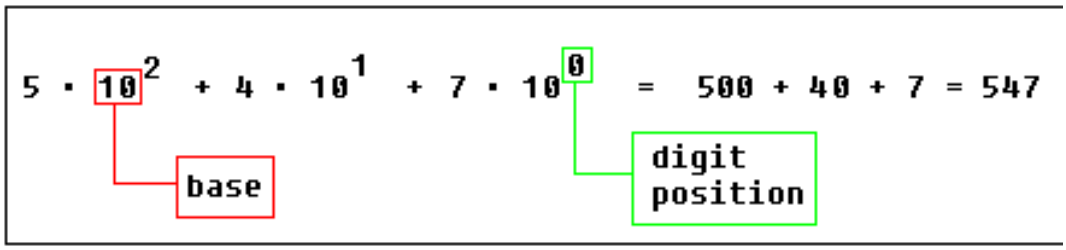
Number System:

Decimal System: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

For example: 754



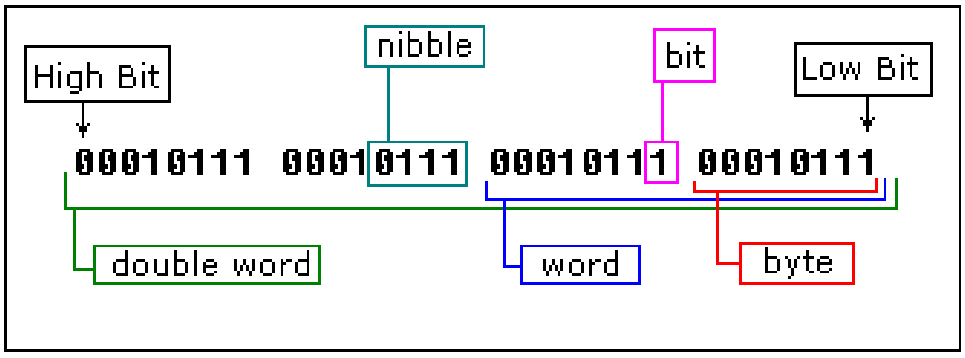
547:



Binary System:

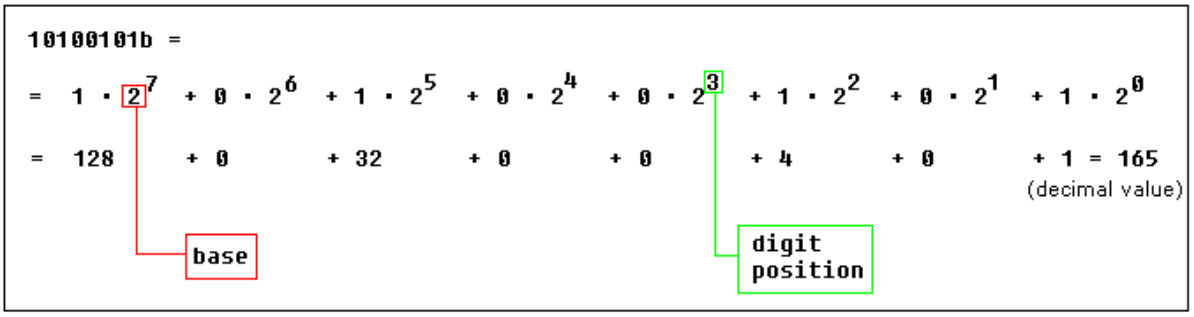
Binary system uses 2 digits: 0, 1

Each digit in a binary number is called a **BIT**, 4 bits form a **NIBBLE**, 8 bits form a **BYTE**, two bytes form a **WORD**, two words form a **DOUBLE WORD** (rarely used):



The binary number **10100101b** equals to decimal value of 165:

ASSEMBLY LANGUAGE LAB 01



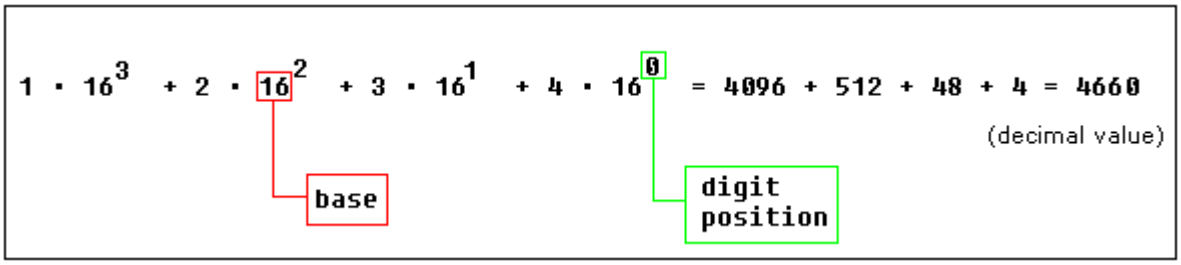
Hexadecimal System:

Hexadecimal System uses 16 digits:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

The hexadecimal number **1234h** is equal to decimal value of 4660:



ASSEMBLY LANGUAGE LAB 01

Numbering Systems Tutorial

What is it?

There are many ways to represent the same numeric value. Long ago, humans used sticks to count, and later learned how to draw pictures of sticks in the ground and eventually on paper. So, the number 5 was first represented as: **| | | | |** (for five sticks).

Later on, the Romans began using different symbols for multiple numbers of sticks:**| | |** still meant three sticks, but a **V** now meant five sticks, and an **X** was used to represent ten of them!

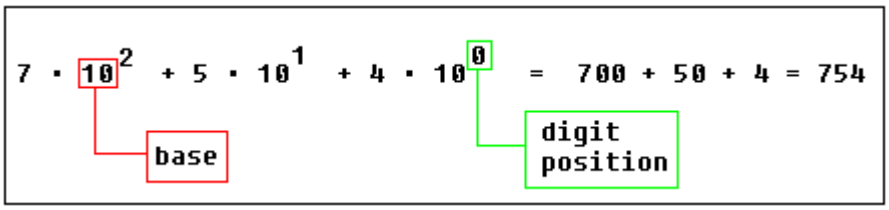
Using sticks to count was a great idea for its time. And using symbols instead of real sticks was much better. One of the best ways to represent a number today is by using the modern decimal system. Why? Because it includes the major breakthrough of using a symbol to represent the idea of counting ***nothing***. About 1500 years ago in India, **zero** (**0**) was first used as a number! It was later used in the Middle East as the Arabic, *sifr*. And was finally introduced to the West as the Latin, *zephiro*. Soon you'll see just how valuable an idea this is for all modern number systems.

Decimal System

Most people today use decimal representation to count. In the decimal system there are 10 digits:

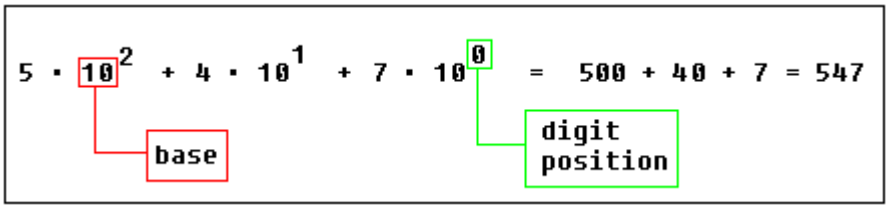
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

These digits can represent any value, for example: **754**.
The value is formed by the sum of each digit, multiplied by the **base** (in this case it is **10**because there are 10 digits in decimal system) in power of digit position (counting from zero):

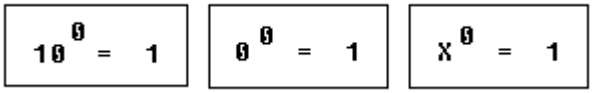


Position of each digit is very important! for example if you place "7" to the end: **547**

it will be another value:



Important note: any number in power of zero is 1, even zero in power of zero is 1:



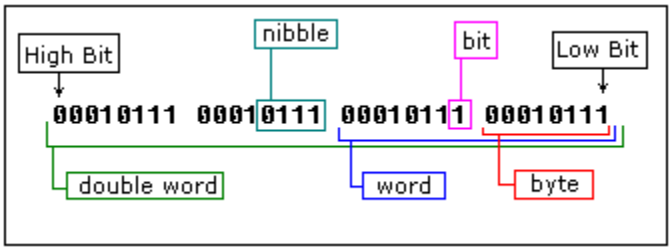
Binary System

Computers are not as smart as humans are (or not yet), it's easy to make an electronic machine with two states: **on** and **off**, or **1** and **0**. Computers use binary system, binary system uses 2 digits:

0, 1

And thus the **base** is **2**.

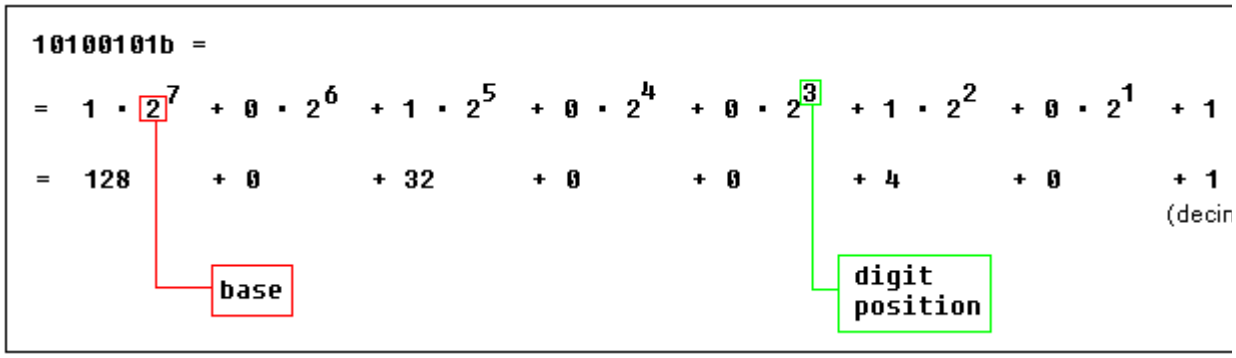
Each digit in a binary number is called a **BIT**, 4 bits form a **NIBBLE**, 8 bits form a **BYTE**, two bytes form a **WORD**, two words form a **DOUBLE WORD** (rarely used):



There is a convention to add **"b"** in the end of a binary number, this way we can determine that 101b is a binary number with decimal value of 5.

The binary number **10100101b** equals to decimal value of 165:

ASSEMBLY LANGUAGE LAB 01



Hexadecimal System

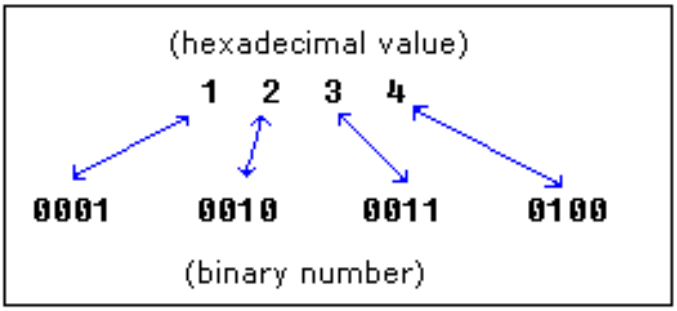
Hexadecimal System uses 16 digits:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

And thus the **base** is **16**.

Hexadecimal numbers are compact and easy to read.
It is very easy to convert numbers from binary system to hexadecimal system and vice-versa, every nibble (4 bits) can be converted to a hexadecimal digit using this table:

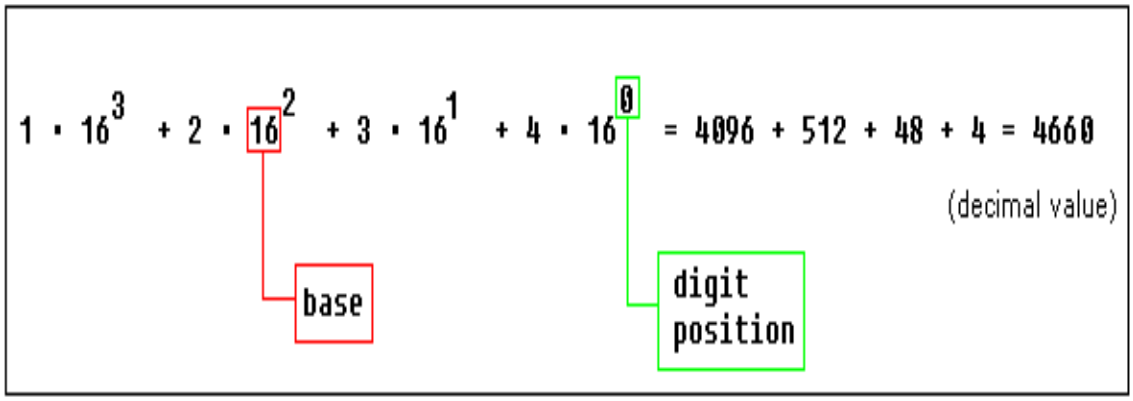
Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A



11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

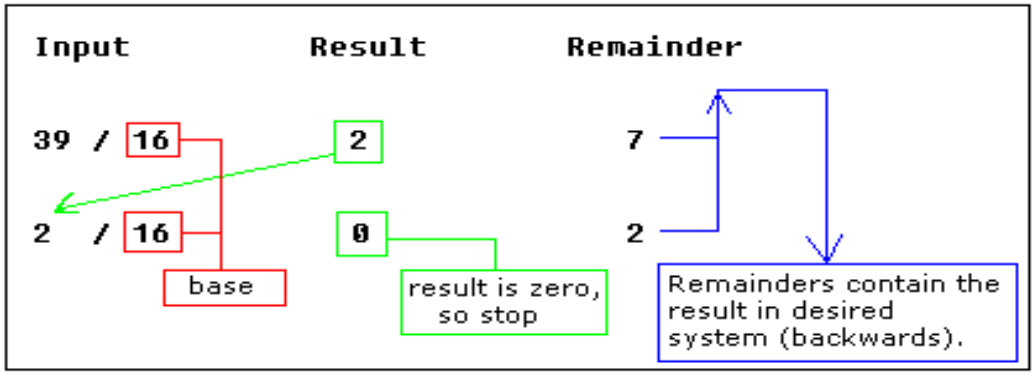
There is a convention to add **"h"** in the end of a hexadecimal number, this way we can determine that 5Fh is a hexadecimal number with decimal value of 95. We also add **"0"** (zero) in the beginning of hexadecimal numbers that begin with a letter (A..F), for example **0E120h**.

The hexadecimal number **1234h** is equal to decimal value of 4660:



Converting from Decimal System to Any Other

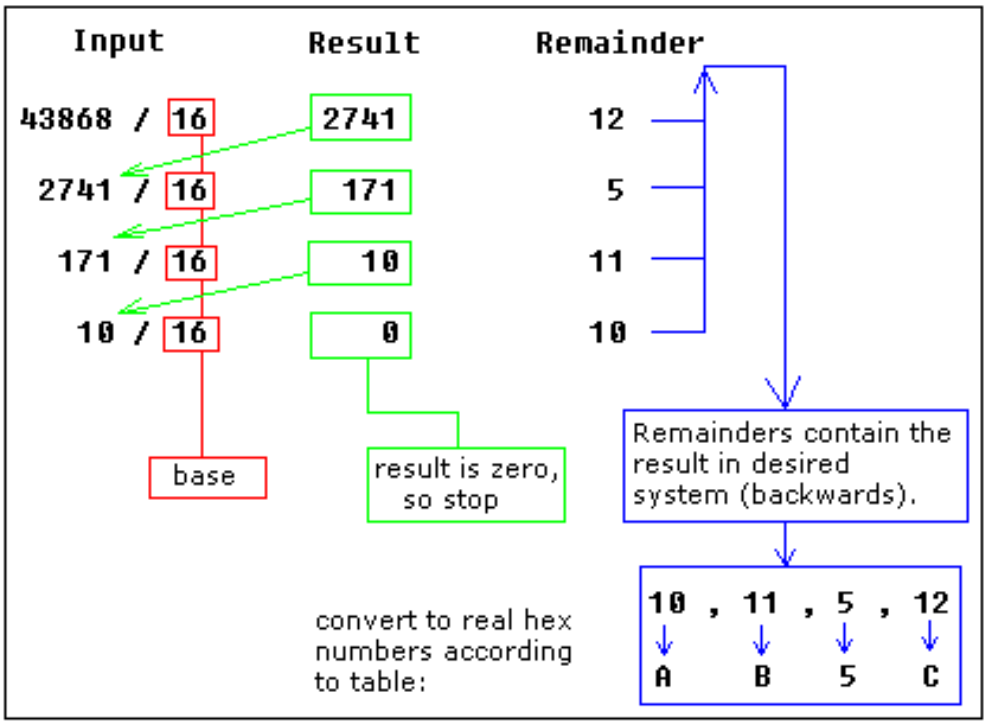
Let's convert the value of **39** (base 10) to *Hexadecimal System* (base 16):



As you see we got this hexadecimal number: **27h**.

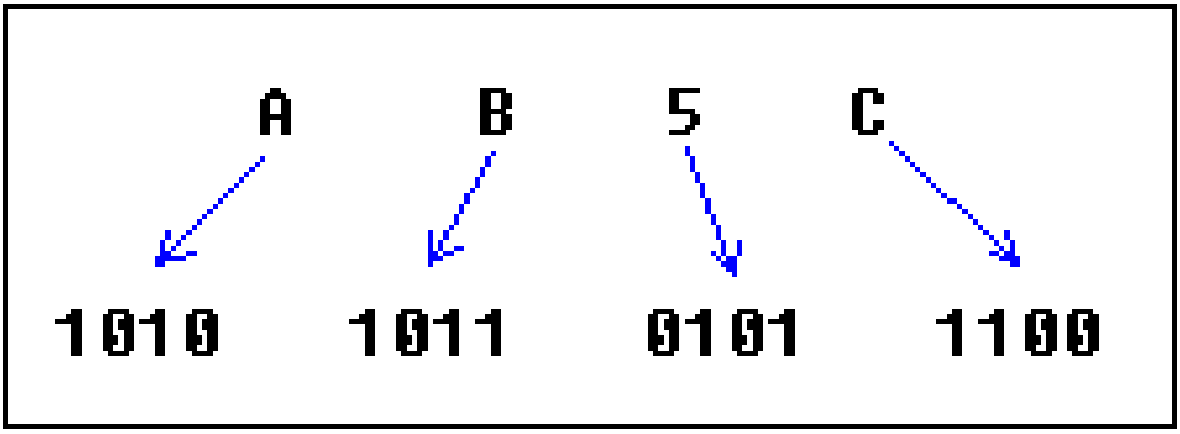
All remainders were below **10** in the above example, so we do not use any letters.

Let us convert decimal number **43868** to hexadecimal form:



The result is **0AB5Ch**,

Convert it to binary number



ASSEMBLY LANGUAGE LAB 01

Signed Numbers:

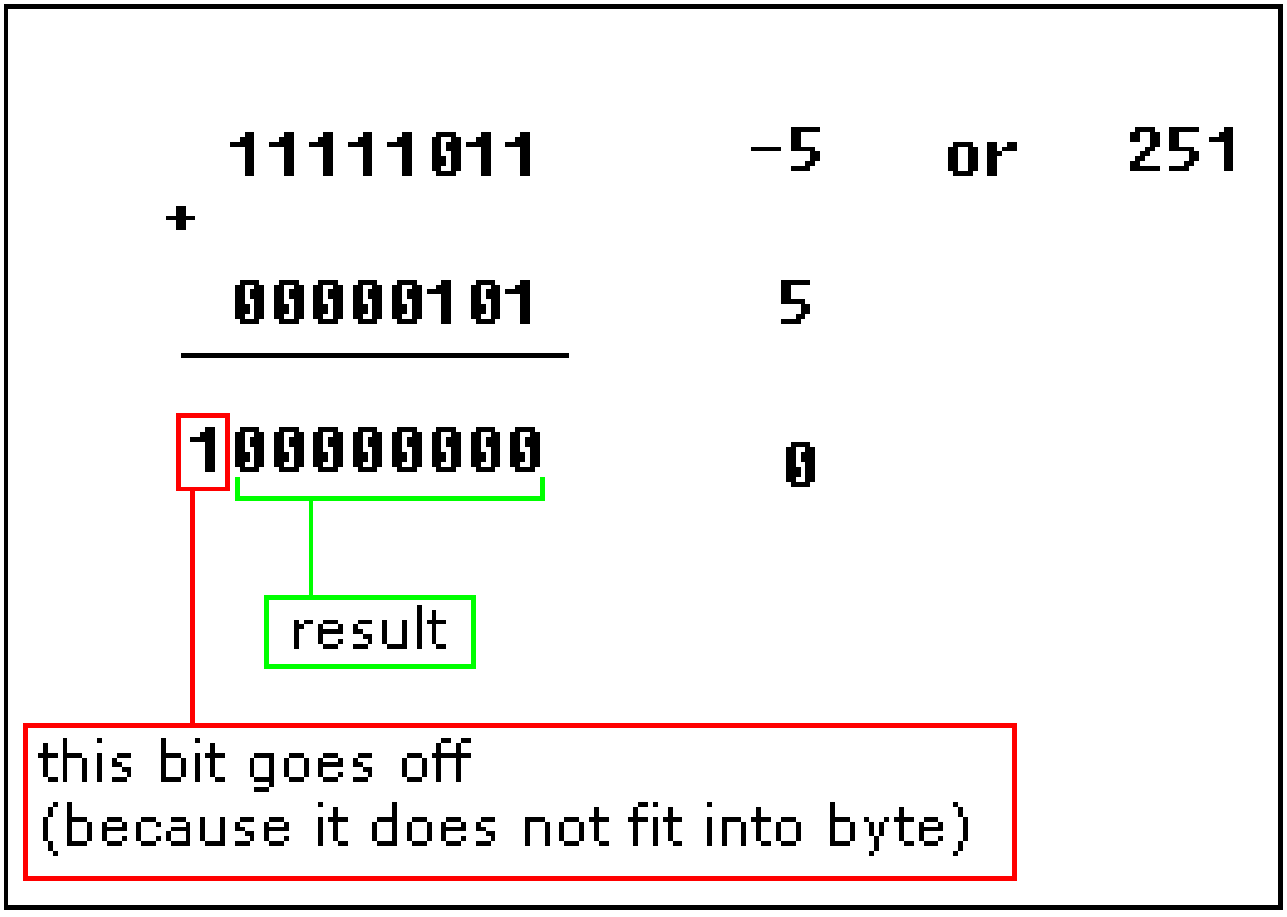
The hexadecimal byte **0FFh** is positive or negative.

It can represent both decimal value "**255**" and "**- 1**"

8 bits can be used to create **256** combinations (including zero), so we simply presume that first**128** combinations (**0..127**) will represent positive numbers and next **128** combinations (**128..256**) will represent negative numbers.

In order to get "**- 5**", we should subtract **5** from the number of combinations (**256**), so it we'll get: **256 - 5 = 251**.

Using this complex way to represent negative numbers has some meaning, in math when you add "**- 5**" to "**5**" you should get zero.



When combinations **128..256** are used the high bit is always **1**, so this maybe used to determine the sign of a number.

ASSEMBLY LANGUAGE LAB 01

Introduction to Debug.exe:

Just Type In google (pakacademy79) or

https://pakacademy79.blogspot.com/p/assemblymasm.html

Debug:

To create a program in assembler two options exist, the first one is to use the assembler program such as MASM or Microsoft Assembler, and the second one is to use the debugger - on this first section we will use this last one since it is found in any PC with the MS-DOS, which makes it available to any user who has access to a machine with these characteristics.

Debug can only create files with a .COM extension, and because of the characteristics of these kinds of programs they cannot be larger than 64 kb, and they also must start with displacement, offset, or 0100H memory direction inside the specific segment.

Debug Reference file Link below

https://pakacademy79.blogspot.com/p/assemblymasm.html